VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

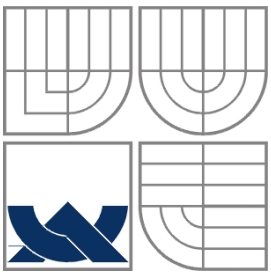
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ZTRÁTOVÁ KOMPRESE OBRAZU

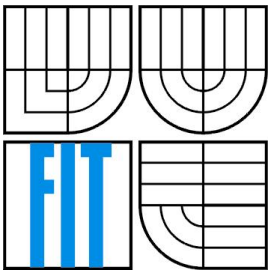
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE DAVID BAŘINA
AUTHOR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ZTRÁTOVÁ KOMPRESE OBRAZU

LOSSY IMAGE COMPRESSION

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

DAVID BAŘINA

VEDOUCÍ PRÁCE
SUPERVISOR

ING. STANISLAV SUMEC, PH.D.

BRNO 2007

Zadání bakalářské práce/5779/2006/xbarin02

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2006/2007

Zadání bakalářské práce

Řešitel: **Bařina David**
Obor: Informační technologie
Téma: **Ztrátová komprese obrazu**
Kategorie: Počítačová grafika

Pokyny:

1. Prostudujte techniky používané pro ztrátovou kompresi obrazu.
2. Navrhněte postup ztrátové komprese obrazu s využitím pokročilých metod.
3. Implementujte knihovnu pro kompresi a dekompresi obrazu pracující na základě algoritmu navrženého v bodu 2.
4. Proveďte porovnání výsledků s běžně používanými formáty využívajících ztrátovou kompresi.
5. Diskutujte dosažené výsledky a možnosti případného pokračovaní práce.

Literatura:

- dle pokynů vedoucího

Při obhajobě semestrální části projektu je požadováno:

- Body 1. a 2.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Sumec Stanislav, Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2006
Datum odevzdání: 15. května 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Pavel Zemčík
vedoucí ústavu

Licenční smlouva

Licenční smlouva je uložena v archivu Fakulty informačních technologií Vysokého učení technického v Brně.

Abstrakt

Tato práce se zabývá pokročilými metodami ztrátové komprese obrazu. Mezi tyto metody patří diskrétní kosinová transformace a Haarova diskrétní vlnková transformace. Dále je zde vysvětlen princip barevných modelů a algoritmů používaných při bezeztrátové kompresi dat. Cílem práce je implementace knihovny pracující na základě těchto poznatků.

Klíčová slova

Komprese obrazu, JPEG, RGB, $Y'C_bC_r$, DCT, DWT, entropické kódování, bzip2.

Abstract

This thesis deals with advanced methods of lossy image compression. To the midst of these methods belong the discrete cosine transform and Haar's discrete wavelet transform. Furthermore principles of color models and algorithms used in lossless data compression are explained in this text.
The purpose of this thesis is the library implementation that is based on these cognizances.

Keywords

Image compression, JPEG, RGB, $Y'C_bC_r$, DCT, DWT, entropy encoding, bzip2.

Citace

David Bařina: Ztrátová komprese obrazu, bakalářská práce, Brno, FIT VUT v Brně, 2007

Ztrátová komprese obrazu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením
Ing. Stanislava Sumce, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
David Bařina
14.5.2007

Poděkování

Rád bych v této sekci poděkoval vedoucímu mé práce za poskytnutou odbornou pomoc.

Obsah

1 Úvod

V dnešní době, kdy jsou dostupné digitální fotoaparáty, má mnoho uživatelů na disku svého počítače velké množství fotografií z nejrůznějších životních událostí. V nekomprimované podobě by tyto fotografie zabíraly zcela neúměrnou diskovou kapacitu a při přenosu by zbytečně zatěžovaly počítačové sítě. Proto vznikly různé metody komprese, které se snaží redukovat objem dat potřebný k uložení takovéhoto obrázku. Dnes nejrozšířenějším standardem pro ztrátovou kompresi obrazu je formát JPEG. Tento standard je z roku 1992. V současné době existuje několik nerozšířených knihoven pro ztrátovou kompresi obrazu, které se snaží jej nahradit a které využívají novější a dokonalejší postupy komprese dat. V této práci se snažím prostudovat různé algoritmy užívané při kompresi obrazu a navrhnou a implementovat vlastní knihovnu, která bude těchto pokročilých metod využívat.

Ve druhé kapitole vysvětluji principy lidského vidění a popisuji první matematicky definovaný barevný model CIE XYZ. Dále popisuji v počítačové grafice široce používaný model RGB a model $Y'C_bC_r$, který je vhodný právě ke ztrátové kompresi obrazu.

Třetí kapitola se zabývá rozdělením komprese obrazu na bezeztrátovou a ztrátovou. Blíže popisuje postup ztrátové komprese. Vysvětluje důvod dekompozice obrazu na menší bloky, převod z modelu RGB na $Y'C_bC_r$ a další zpracování transformací na frekvenční koeficienty.

Ke transformaci na frekvenční koeficienty se používají buď metody založené na Fourierově transformaci nebo metody založené na transformaci waveletové, česky vlnkové. Na těchto frekvenčních koeficientech se provede cílená ztráta dat tak, aby lidský zrak tuto změnu co nejméně postřehl. Koeficienty je třeba převést na tok čísel, které budou dále zpracovávány některou bezeztrátovou kompresní metodou.

Další kapitola se věnuje těmto bezeztrátovým metodám. Popisuje také knihovnu libbzip2, kterou jsem se rozhodl pro kompresi koeficientů ve své knihovně použít já.

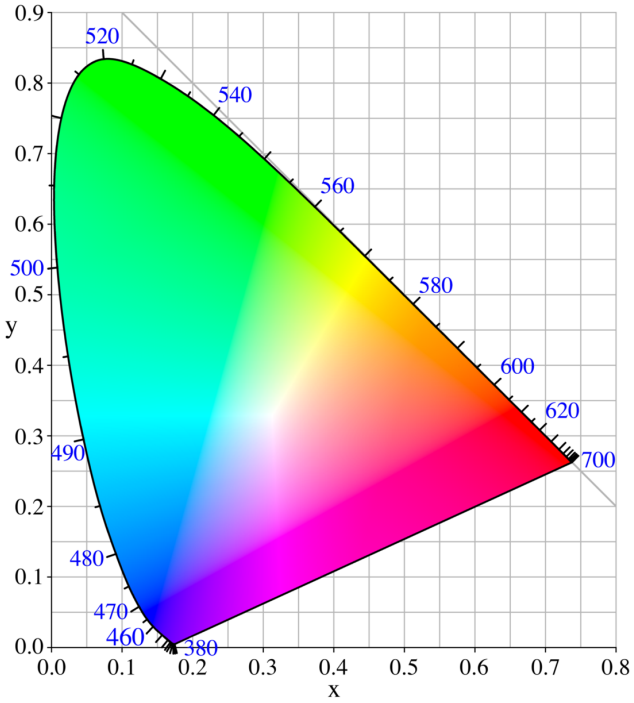
V kapitole 6 je podrobně rozebrána vlastní implementace mé knihovny, postup komprese a poté dekomprese. Dále zde porovnávám efektivnost mnou implementovaných metod a formátu JPEG.

2 Barevné modely

Člověk vnímá okolní svět mimo jiné očima. Princip lidského zraku je ve zkratce následující. Světlo dopadající skrze čočku na sítnici způsobuje chemické změny ve světlocitlivých buňkách (receptorech) nazývaných tyčinky a čípky. Tyto pak vysílají nervové impulsy do mozku, který si z nich poskládá viděný obraz. Tyčinky jsou citlivé na intenzitu světla (vnímají stupně šedi). Čípky jsou trojího druhu a umožňují barevné vidění - vnímají červenou, zelenou a modrou složku světla.

2.1 CIE XYZ

Barva je odvozena z viditelného spektra světla (vlnové délky) a popisuje vlastnost tohoto světla z hlediska vnímání lidským okem. Lze ji reprezentovat mnoha modely. V roce 1931 vytvořil mezinárodní úřad *Commission internationale de l'éclairage* (zkráceně CIE) studii o vnímání barev. V této studii byl poprvé matematicky definován model barev nazvaný CIE XYZ (nebo také CIE 1931). Tento barevný model vycházel z předchozí specifikace zvané CIE RGB (viz. [5]), která vzešla ze série experimentů. Tyto dva barevné modely lze vzájemně převádět. Protože lidské oko má 3 typy senzorů barev, bylo by znázornění všech viditelných barev trojrozměrným diagramem. Koncept vnímání barev ovšem může být rozdělen na 2 části – jas (vnímají tyčinky) a barvu (vnímají čípky). Model CIE XYZ byl záměrně navržen tak, že parametr Y vyjadřuje jas a vlastní barva je specifikována dvěma odvozenými parametry *x* a *y*. Tyto odvozené parametry lze spočítat ze všech tří trichromatických složek X, Y a Z. Barevný model udaný parametry *x* a *y* se nazývá CIE xyY nebo jen CIE xy (bez parametru jasu). Tento barevný model je znázorněn chromatickým diagramem na obrázku číslo 1.



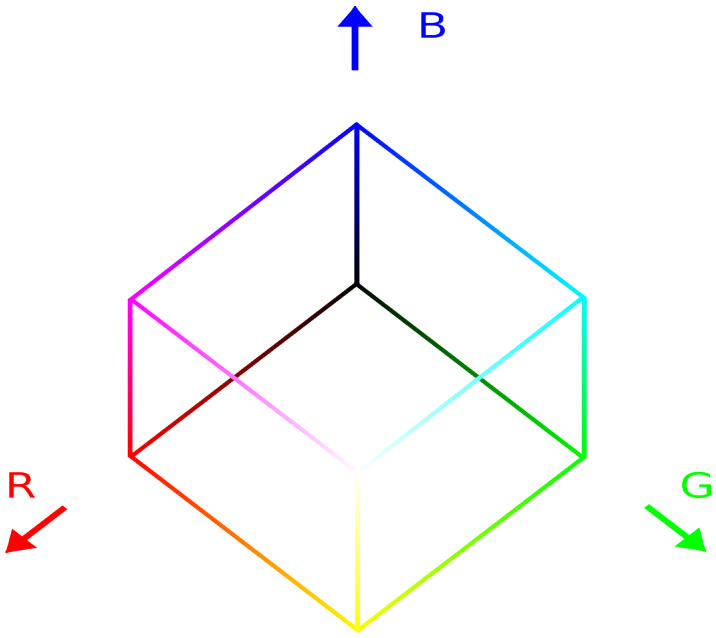
Obrázek 1: Chromatický diagram barevného modelu CIE 1931 (x, y = nanometry)

2.2 RGB

RGB je zkratka z anglických slov *red*, *green* a *blue* – česky červená, zelená a modrá. Tímto barevným modelem se popisují barvy pro CRT monitory a obraz v počítači je tudíž nejčastěji reprezentován právě tímto modelem.

Tyto tři výše zmíněné barvy se nazývají barvy primární. Jejich kombinací při vyzařování lze poskládat barvy jiné (aditivní skládání barev). Pro všechny parametry (R, G a B) platí, že nejnižší hodnota (0) znamená nejnižší podíl světla dané vlnové délky ve výsledné barvě. Všechny tři parametry nulové vyjadřují černou (nic nevyzařuje). Naopak všechny tři na maximální intenzitě (1) znamenají bílou. Tento barevný model lze znázornit tzv. jednotkovou krychlí – viz. obrázek číslo 2.

Barevný model používaný v počítačové grafice neodpovídá přímo CIE 1931 (CIE RGB). Neexistuje dokonce jediný barevný model RGB. Nejčastěji používané RGB modely jsou Adobe RGB (digitální kamery) a především sRGB (počítačové CRT monitory). Konkrétní model RGB (správně R'G'B' pro odlišení od CIE RGB) je specifikován definicí tří primárních barev a tzv. bílým bodem. Primární barvy se definují chromatickými souřadnicemi barvy z CIE xy. Bílý bod stejným způsobem definuje bílou barvu.



Obrázek 2: Jednotková krychle barevného modelu RGB

V dnešních počítačích se pro obraz v modelu RGB typicky používá 24 bitů na jeden pixel. To znamená, že na každou složku připadá 8 bitů, tedy 256 úrovní. Celkem lze tedy vyjádřit 16 777 216 různých barev (dáno kombinací 256^3).

2.3 $Y'C_bC_r$

Barevný model RGB se ke ztrátové kompresi obrazu příliš nehodí, protože hodnoty složek R, G, B se mezi sousedními pixely často mění. Změna jedné hodnoty navíc ovlivní nejen barevnou složku ale také jas bodu a lidský zrak nejvíce vnímá právě jas. Z tohoto důvodu je vhodnější pixel reprezentovat jako jas (v modelu CIE 1931 odpovídá parametru Y) a další složky, které udají vlastní

barvu. U složek udávajících barvu (anglicky se nazývají *chrominance*) se je pak při kompresi možné dopustit větší nepřesnosti než u jasu (anglicky *brightness* nebo *luminance*), protože to lidský zrak hůře rozpozná.

Dle standardu ITU-R BT.601 (dříve CCIR Recommendation 601 nebo jen CCIR 601) od organizace *Comité consultatif international pour la radio* se jas (intenzita) pro digitální signál spočítá dle vzorce:

$$Y' = 0.299 \cdot R' + 0.587 \cdot G' + 0.114 \cdot B'$$

Dle dokumentu [1] by se tato intenzita měla nazývat *luma* a ne *luminance*, protože je spočítána z upravených R'G'B' a nikoli z CIE RGB. Barevnou složku (*chrominance*) lze udat jako rozdíl primární barvy (typicky modré a červené) od tohoto jasu. Tyto parametry se označují jako C_b (*blue*) a C_r. (*red*). Pro počítačovou grafiku (digitální signály) by se podle výše zmíněného dokumentu měl používat pojem *chroma*. Tento barevný systém se správně označuje jako $Y'C_bC_r$. Často se však lze setkat se zažitým ale nesprávným označením YC_bC_r. Stejně jako u RGB i u tohoto modelu existuje několik variant. Jednou z nejrozšířenějších variant je varianta označovaná jako JPEG-YC_bC_r, jež je definována ve specifikaci JFIF [2]. Vychází ze standardu BT.601 (tedy upravených R', G', B') a všechny 3 parametry vyjadřuje na 8 bitech (256 úrovní), což je vhodné pro počítačové zpracování.

3 Komprese obrazu

Komprese dat je postup mající za cíl zmenšit objem dat nebo jiných zdrojů nutný k uložení dané informace. Komprese obrazu má tedy za úkol zmenšit objem dat reprezentující daný obraz. Kompresi rastrového obrazu můžeme v zásadě rozdělit na bezeztrátovou a ztrátovou. Kompresním poměrem označujeme poměr zkomprimovaných dat k datům původním.

Při bezeztrátové kompresi obrazu nedochází ke ztrátě informací. Rekonstruovaný obraz tedy přesně odpovídá obrazu nekomprimovanému. Bezeztrátovou kompresí se však prakticky nedosahuje takového kompresního poměru jako kompresí ztrátovou.

Naopak při ztrátové kompresi obrazu dochází ke ztrátě některých informací, takže nelze ze zkomprimovaných dat přesně vytvořit původní obraz. Lidské vidění ovšem není dokonalé a je méně citlivé na některé komponenty obrazu než na jiné. V ideálním případě nedokáže člověk bez detailního zkoumání rozlišit původní obraz od obrazu poškozeného ztrátou informací způsobenou kompresí.

3.1 Bezeztrátová komprese obrazu

Postupy pro bezeztrátovou kompresi většinou používají Huffmanova nebo aritmetického entropického kódování. Entropické kódování přiřazuje symbolům bitové kódy podle pravděpodobnosti výskytu symbolu. Dále se snaží o predikci právě kódovaného pixelu z pixelů předcházejících. Při této kompresi nedochází k převodu barevného modelu obrazu na jiný, jak je tomu běžné u ztrátové komprese.

3.1.1 Grafický formát PNG

Typickým příkladem bezeztrátového formátu pro kompresi obrazu je formát PNG (zkratka z anglického *Portable Network Graphics)*.

Tento formát umožňuje uložit obraz ve stupních šedi (256 úrovní), barevný obraz reprezentovaný složkami RGB (24 bitová hloubka) nebo RGBA (RGB s 8bitovým alfa kanálem). Obraz může být kódován postupně (progresivně) nebo řádkově prokládaně. Ke kompresi používá algoritmu DEFLATE (kombinace algoritmu LZ77 a Huffmanova kódování).

3.2 Ztrátová komprese obrazu

Postupy pro ztrátovou kompresi obrazu využívají toho, že lidské oko je méně citlivé na barevné složky obrazu než na intenzitu pixelů, a toho, že člověk vnímá méně vysoké frekvence obsažené v obrazu než frekvence nízké. Při ztrátové kompresi tedy dochází ke ztrátě těch informací, na které

je lidský zrak méně citlivý. Ztráta těchto informací je výhodná, protože dochází k výraznému zmenšení zkomprimovaných dat.

Při kompresi obrazu jež je reprezentován barevnými komponentami R, G, B (správně R', G', B') dochází typicky ke konverzi na jiný barevný model, nejčastěji $Y'C_bC_r$. Následně je provedena transformace nebo přeskupení dat tak, aby bylo jednoduše možné oddělit důležité informace, na které je lidské vidění citlivé, od méně důležitých. Méně důležité informace jsou pak uloženy s menší přesností nebo vůbec. Tím dochází ke ztrátě informací a ke zmenšení objemu výsledných dat. Data jsou dále zpracována některou bezeztrátovou metodou (např. pomocí Huffmanova kódování).

3.2.1 Konverze barevného modelu

V podkapitole 2.3 jsem uvedl, že nejrozšířenější varianta barevného modelu $Y'C_bC_r$ je varianta označovaná jako JPEG-YC_bC_r, (definována ve specifikaci JFIF [2]). V této specifikaci jsou uvedeny rovnice pro převod barvy v modelu RGB na dané YC_bC_r, a zpět. Tyto rovnice očekávají 8 bitové složky R, G, B (hodnoty v intervalu 0 až 255) a produkují taktéž 8 bitové složky Y, C_b, C_r v tomtéž intervalu. Tyto složky by se měli nazývat *luma* (Y) a *chroma* (C_b, C_r). Zmíněné rovnice vypadají následovně.

Pro převod RGB na YC_bC_r:

$$\begin{aligned} Y &= 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \\ C_b &= -0.1687 \cdot R - 0.3313 \cdot G + 0.5 \cdot B + 128 \\ C_r &= 0.5 \cdot R - 0.4187 \cdot G - 0.0813 \cdot B + 128 \end{aligned} \tag{1}$$

Pro převod YC_bC_r na RGB:

$$\begin{aligned} R &= Y + 1.402 \cdot (C_r - 128) \\ G &= Y - 0.34414 \cdot (C_b - 128) - 0.71414 \cdot (C_r - 128) \\ B &= Y + 1.772 \cdot (C_b - 128) \end{aligned} \tag{2}$$

Protože při výpočtu dochází k zaokrouhlování na celá čísla, je již tento převod ztrátový. Výsledkem převodu obrazu z RGB do YC_bC_r jsou 3 samostatné složky obrazu, které je možné dále komprimovat odděleně. Intenzita Y se často komprimuje jinou metodikou než C_b a C_r, protože lidské oko je více citlivé na jas než na barvu. Složky C_b a C_r, se mění velmi mírně a jsou blízké nule. Jsou proto pro kompresi výhodné.

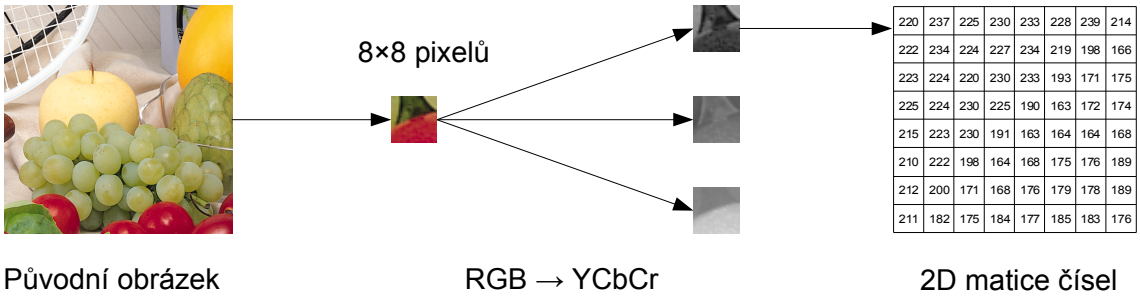
3.2.2 Metody transformace

Úkolem transformace je převést obrazová data do takové formy, u níž je možné jednoduše určit důležitost (z hlediska vnímání lidským zrakem) obrazové informace.

Obraz je v podstatě dvourozměrná matice čísel (mající určitou výšku a šířku). Pokud je obraz převeden do modelu YC_bC_r, je nyní udán třemi složkami, na které se bude transformace aplikovat odděleně.

Typickým příkladem obrazu uchovávanému v digitální podobě může být fotografie. Poněvadž takovýto obrázek může mít rozměry (výška, šířka) v řádech stovek až tisíců pixelů (celkem tedy miliony pixelů), vyžadovalo by zpracování obrázku jako celku vysoký výpočetní výkon. Proto je obraz rozdělován na menší úseky, tzv. bloky, které jsou zpracovávány samostatně. Volba rozměrů bloku může záviset i na zvoleném algoritmu transformace. Dnes často používanou hodnotou je rozměr 8×8 pixelů (bodů obrazu). Tento rozměr je použit např. ve formátu JPEG a formátech MPEG.

Obraz je matice neboli dvourozměrné pole čísel (složek Y, C_b, a C_r). Bod obrazu v tomto poli lze určit pomocí souřadnic x a y. Obraz lze také považovat za dvourozměrnou funkci, funkci mající 2 parametry – x, y – a vracející hodnotu obrazového bodu v daném místě. V digitálním světě je tato funkce či signál je diskrétní. Funkci lze pomocí jedné z vhodných transformací rozložit na frekvenční složky. Hodnoty vysokofrekvenčních složek bude následně možné znepřesnit nebo vypustit zcela. Celý postup takovéto dekompozice obrazu znázorňuje obrázek číslo 3.



Obrázek 3: Dekompozice obrazu

Často používanou transformací je tzv. diskrétní kosinová transformace (zkráceně DCT), jež je příbuzná diskrétní Fourierově transformaci. Fourierova transformace rozloží signál na jednotlivé frekvence. DCT je použita např. ve formátech JPEG (viz. [3]) a MPEG.

Další používanou transformací může být diskrétní vlnková transformace, kterou lze chápat zjednodušeně tak, že rozdělí signál na 2 související signály a opakovaně se aplikuje na jeden z nich. Tato metoda je použita například ve formátu JPEG 2000.

4 Funkcionální analýza

Obraz je v podstatě dvourozměrná diskrétní funkce. Její analýzou pomocí vhodné transformace se rozloží na koeficienty reprezentující jisté složky obrazu. Obraz je funkce (matice) dvourozměrná a její transformací vznikne tedy jiná dvourozměrná matice. Pro zápis koeficientů do toku dat je potřeba tuto dvourozměrnou matici linearizovat na jednorozměrný vektor.

4.1 Diskrétní kosinová transformace

Diskrétní kosinová transformace (anglicky *discrete cosine transform*, zkráceně DCT) je diskrétní transformace podobná diskrétní Fourierově transformaci (DFT), ale na rozdíl od ní produkující pouze reálné koeficienty (viz. [6]). Je jednou z mnoha transformací příbuzných Fourierově transformaci.

Formálně je DCT lineární invertovatelná funkce, která transformuje vektor (pole) N reálných čísel $x_0, \ldots, x_{N-1}$ na vektor N reálných čísel $X_0, \ldots, X_{N-1}$ zvaných koeficienty. Existuje několik mírně lišících se variant této transformace. Nejběžnější variantou je:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot \cos\left[\frac{\pi}{N} \cdot \left(n + \frac{1}{2}\right) \cdot k\right] \tag{3}$$

Inverzní transformace (IDCT) k této variantě je následující funkce násobená *2/N*:

$$x_k = \frac{1}{2} \cdot X_0 + \sum_{n=1}^{N-1} X_n \cdot \cos\left[\frac{\pi}{N} \cdot \left(k + \frac{1}{2}\right) \cdot n\right] \tag{4}$$

Tyto transformace jsou aplikovatelné na jednorozměrný signál, obraz je však dvourozměrný. Vícerozměrná transformace (transformace vícerozměrné funkce) může být spočítána jako série jednorozměrných transformací postupně v každém rozměru. Pro dvourozměrnou verzi nejprve po řádcích a pak po sloupcích (nebo naopak). Dvourozměrná DCT-je například dána rovnicí:

$$X_{k_1, k_2} = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \cos\left[\frac{\pi}{N_1} \cdot \left(n_1 + \frac{1}{2}\right) \cdot k_1\right] \cos\left[\frac{\pi}{N_2} \cdot \left(n_2 + \frac{1}{2}\right) \cdot k_2\right]$$

V případě jednorozměrné verze je vyprodukován nejprve koeficient, označovaný jako stejnosměrná složka signálu (označována jako DC z anglického *direct current*), a následně série koeficientů udávajících obsah bázové funkce (kosinus) v signálu. Každý další koeficient vyjadřuje vyšší frekvenci (průběh kosinu se zopakuje vícekrát oproti předchozímu koeficientu).

V případě dvourozměrné verze je vyprodukována matice koeficientů. Koeficient s indexem *(0, 0)* vyjadřuje opět stejnosměrnou složku (DC). Další koeficienty vzdalující se od tohoto horizontálně či vertikálně vyjadřují obsah bázové funkce s odpovídajícími horizontálně a vertikálně znásobenými frekvencemi.

Původní funkci lze z koeficientů zrekonstruovat beze ztráty dat. Na rozdíl od diskrétní Fourierovy transformace jsou nejvýznamnější koeficienty nahrnuty k počátku matice (index *(0, 0)*).

S rostoucí vzdáleností od prvního (DC) koeficientu je každý další méně významný a lze na něm provést větší ztrátu informace (například více zaokrouhlit).

4.2 Diskrétní vlnková transformace

Diskrétní vlnková transformace (anglicky *discrete wavelet transform*, zkratkou DWT) je transformace odvozená z vlnkové transformace pro diskrétní vlnky (anglicky *wavelety*). Vlnka je matematická funkce používaná k rozdělení jiné funkce na frekvenční komponenty. Vlnková transformace je transformace definovaná jako konvoluce mateřské vlnky a vstupního signálu.

Konvoluce dvou diskrétní signálů se značí znakem * a je definována vztahem:

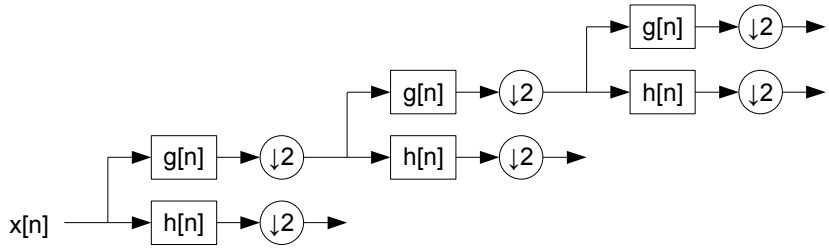
$$(f * g)[m] = \sum_{n} f[n] \cdot g[m-n]$$

Provedení DWT signálu *x* lze chápat jako průchod skrze sérii filtrů. Nejprve se vzorky nechají projít skrze dolní propusť s impulzní odezvou *g*. Signál je také současně dekomponován použitím filtru horní propusti *h*. Výstupy dávají podrobné koeficienty (z horní propusti) a přibližné koeficienty (z dolní propusti). Je důležité, že tyto dva filtry spolu vzájemně souvisejí. Filtry jsou popsány následujícími rovnicemi:

$$y_{low}[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot g[2 \cdot n - k]$$

$$y_{high}[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[2 \cdot n - k]$$

Výstupy filtru jsou dále podvzorkovány dvěma. Počítání celé konvoluce *x* * *g* s následným podvzorkováním může mrhat výpočetním časem. Optimalizace, kde jsou tyto 2 výpočty prokládány se označuje jako *Lifting scheme*. Tato dekompozice je opakována k dalšímu zvýšení frekvenčního rozlišení. To je reprezentováno jako binární strom s uzly reprezentujícími podprostor s rozdílným umístěním času/frekvence. Strom je znám jako banka filtrů:



Obrázek 4: Banka filtrů pro DWT

Kvůli procesu dekompozice musí mít vstupní signál délku 2^n, kde *n* je počet stupňů. Vícerozměrná transformace (obraz) může být opět jako u DCT spočítána jako série jednorozměrných transformací postupně v každém rozměru. Výsledkem takovéhoto výpočtu je opět matice koeficientů. Koeficient s indexem (0, 0) reprezentuje průměrnou hodnotu vstupního signálu. Ostatní koeficienty

jsou upřesňující rozdíly (čím vzdálenější od koeficientu (0, 0), tím vyšší frekvence). Ekvivalentně jako u DCT jsou tyto s rostoucí vzdáleností méně významné. Vice v [7].

4.2.1 Haarova vlnková transformace

Pro vstup reprezentovaný vektorem 2^n čísel je Haarova vlnková (waveletová) transformace nejjednodušší spárování vstupních hodnot - uložením rozdílu a předáním součtu (do dalšího stupně transformace). Tento proces je opakován rekurzivně (na součty). Konečný výsledek transformace je 2^n-1 rozdílů a jeden celkový součet. Transformace byla objevena maďarským matematikem jménem Alfréd Haar.

Z programového hlediska se jedná o jeden součet dělený dvěma a jeden rozdíl dělený dvěma pro každý koeficient v jednom stupni transformace. Inverzní transformace se pak jednoduše spočítá jako tento průměrný součet plus průměrný rozdíl a průměrný součet minus průměrný rozdíl.

4.3 Zpracování koeficientů

Výstupem výše zmíněných transformací je matice koeficientů. Tyto koeficienty reprezentují výskyt odpovídající frekvence v původním bloku obrazu. Protože lidské oko je méně citlivé na vysoké frekvence než na nízké a protože je asi dvakrát více citlivé na horizontální změny než na vertikální, lze se na určitých koeficientech dopustit větší ztráty přesnosti než na jiných. Koeficienty jsou čísla reálná. Pro další zpracování bezeztrátovými algoritmy je potřeba je převést na čísla celá. Také je nutné z dvourozměrné matice udělat jednorozměrný vektor, který může být těmto algoritmům předán. Zaokrouhlení na celá čísla je již samo o sobě ztrátou dat. Tomuto postupu se říká kvantizace. Zápis dvourozměrného pole do pole jednorozměrného se nazývá linearizace.

4.3.1 Kvantizace

Každý koeficient by bylo možné pouze zaokrouhlit na nejbližší celé číslo. Tím by sice docházelo jen k velmi malé ztrátě dat, ale velikost dat by se nezmenšila nebo by ještě narostla (počet bitů, na kterém lze vyjádřit koeficient jako celé číslo může být vyšší než původních 8 bitů). Jednotlivá čísla by se od sebe také lišila, což není vhodné pro další postup komprese. Dělením každého koeficientu vhodně zvoleným číslem a následným zaokrouhlením by se situace zlepšila. Zmenšil by se počet bitů, na kterém lze koeficient vyjádřit. Koeficienty s původně blízkou hodnotou by po kvantizaci měly hodnotu stejnou, což je vhodné pro další postup. Docházelo by však u všech frekvencí ke stejné ztrátě dat. Se znalostí vlastností lidského vidění (tzv. psychovizuální model) lze určit pro každý koeficient v matici číslo, kterým bude poděleno. Nízké frekvence budou poděleny nízkým číslem, aby došlo k minimální ztrátě dat. Vysoké frekvence vyšším číslem, které odpovídá pozici koeficientu v matici. Tato čísla lze zvětšovat, respektive zmenšovat pro nižší, respektive vyšší kvalitu komprese. Při nižší kvalitě bude více koeficientů stejných, pravděpodobně nulových,

ale objem dat nutný k popsání obrazu bude značně menší. Matice čísel, kterými mají být odpovídající koeficienty při kvantizaci děleny, se nazývá kvantizační matice. Protože na koeficientech získaných ze složek C_b a C_r je možné se dopustit větší ztráty dat (viz. kapitola 3.2.1), je pro ně používána jiná kvantizační matice než pro koeficienty získané ze složky Y.

Nejjednodušší kvantizační matice mají u koeficientu s indexem *(0, 0)* nízké číslo v řádu jednotek, s přibývající vzdáleností od této pozice kvantizační koeficient roste. Heuristicky však byly odvozeny mnohem účinnější kvantizační matice. Příkladem mohou být výchozí matice pro koeficienty z DCT použité ve standardu JPEG (viz. [3]). Příklad jednoduché matice a matice pro složku luma a složky chroma ze standardu JPEG jsou uvedeny na následujícím obrázku.

4	3	3	4	6	7	8	10
3	3	3	4	5	6	8	10
3	3	3	4	6	9	12	12
4	4	4	7	9	12	12	17
6	5	6	9	12	13	17	20
7	6	9	12	13	17	20	20
8	8	12	12	17	20	20	20
10	10	12	17	20	20	20	20

jednoduchá

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

JPEG - Y

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

JPEG – C_b a C_r

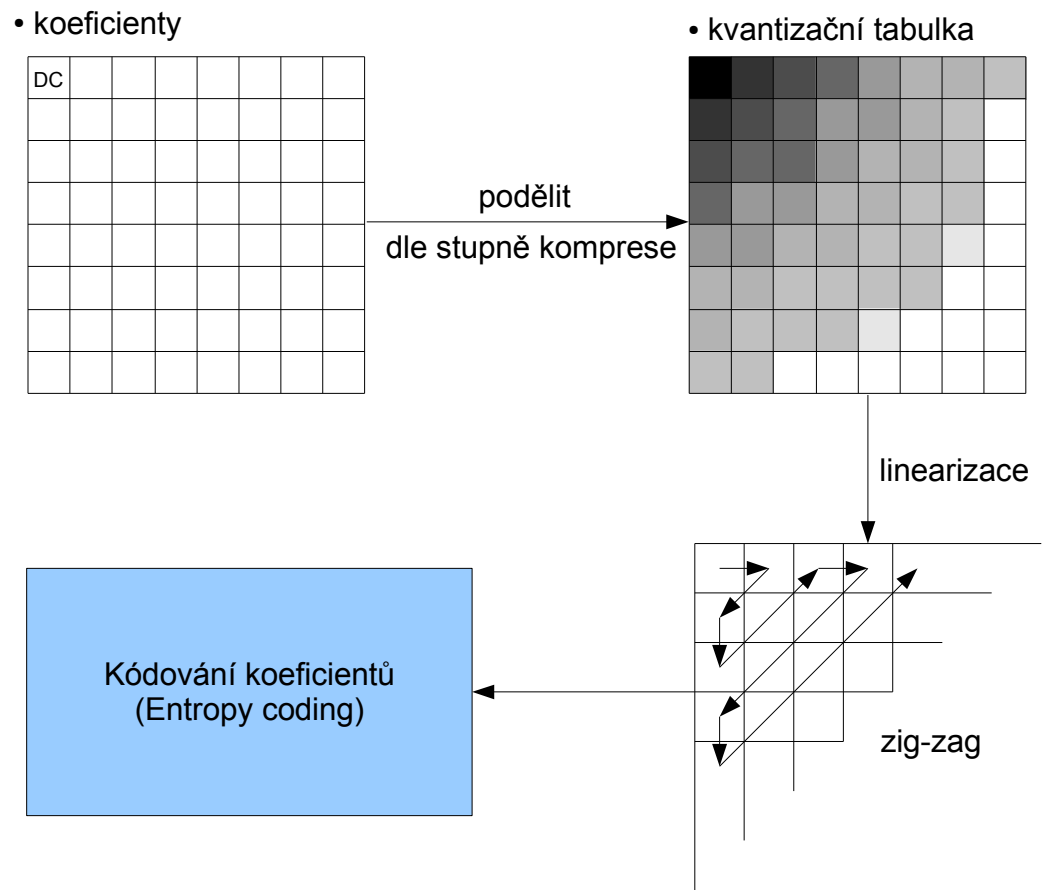
Obrázek 5: Kvantizační matice

Koeficienty v kvantizační matici mohou být pozměněny (například vynásobeny číslem) pro snížení nebo zvýšení kvality komprese (s ní koresponduje velikost zkomprimovaných dat).

4.3.2 Linearizace

Matice koeficientů, na nichž byla provedena takováto kvantizace, bude mít nejvyšší hodnotu u DC koeficientu a nejnižší hodnoty (často 0) v pravém dolním rohu. Této vlastnosti se využívá při linearizaci. Kdyby se takováto matice zapsala do jednorozměrného pole jednoduše po řádcích, neležely by tyto nulové koeficienty za sebou, což není vhodné pro další kompresi. Diagonálním zápisem, kterému se anglicky říká *zig-zag scanning* lze tyto stejné hodnoty umístit do pole za sebe. Matice se začíná procházet od DC koeficientu, postupně se skenují nízké a pak vyšší frekvence a končí se v pravém dolním rohu. Pokud jsou na konci takto zapsaného pole samé nuly, nemusejí se do výstupního toku dat zapisovat a místo nich se zapíše speciální symbol ukončující kódování bloku. Tak linearizaci implementuje například formát JPEG.

Na následujícím obrázku je znázorněn postup komprese od matice koeficientů, její kvantizaci, linearizaci, *zig-zag* průchod maticí, až po předání bezeztrátovým metodám, které zapíší koeficienty přímo do výstupního toku dat.



Obrázek 6: Kvantizace a linearizace koeficientů

5 Entropické kódování

Entropický kodér má za úkol nahradit symboly (tj. vstupní koeficienty) za bitové sekvence tak, že nejčastěji se vyskytujícím symbolům přiřadí nejkratší sekvence a méně častým sekvence delší. Dvě nejpoužívanější metody entropického kódování je Huffmanovo kódování a aritmetické kódování. Kódovacích schémat však existuje celá řada (příkladem unární nebo Fibonacciho kódování).

Příchozí koeficienty obsahují často sekvence po sobě jdoucích stejných čísel. Ty lze výhodně zapsat tzv. RLE kódováním. Koeficienty lze také některou vhodně zvolenou transformací přeskupit tak, aby bylo pravděpodobnější, že stejná čísla za sebou následují.

5.1 Metody kódování

Následující podkapitoly popisují různé metody, které je možno použít při kódovaní koeficientů.

5.1.1 RLE

RLE je zkratka z anglického *Run-length encoding*, což volně přeloženo znamená kódování pomocí délky sledů dat. Jedná se bezeztrátovou metodu komprese dat. Princip spočívá v tom, že kóduje za sebou jsoucí sledy stejných symbolů do dvojic (počet, symbol). Metoda je účinnější na datech, která obsahují větší množství sekvencí stejných symbolů. Příklad komprese:

Vstupní proud:	AAAABBB
Výstupní proud:	(4,A) (3,B)

Tabulka 1: Příklad metody RLE

5.1.2 Burrows-Wheelerova transformace

Anglicky *Burrows-Wheeler transform* (BWT) je pomocný algoritmus používaný v technikách komprese dat. Samotná transformace data nijak nekomprimuje. Způsobí pouze změnu pořadí symbolů (permutaci). V případě, že vstupní řetězec symbolů má několik podřetězců, které se v něm vyskytují vícekrát. Bude mít výstupní řetězec několik míst, kde se vyskytuje stejný symbol několikrát za sebou. To je výhodné například pro RLE kompresi. Transformace ovšem přidává k výstupnímu řetězci informaci o umístění symbolu konce původního řetězce (tzv. EOF symbol).

Princip této transformace spočívá v tom, že se ze vstupního řetězce (zakončeného symbolem EOF) vytvoří všechny jeho možné rotace. Tyto se dále klasicky setřídí. Ze setříděného pole rotací původního řetězce se do výstupního zapíše postupně od počátku poslední symbol z každé rotace. Na výstupu je tedy transformovaný vstup rozšířený o ukazatel na konec původního řetězce. Příklad transformace textového řetězce "^BANANA@" (zavináč značí EOF symbol) je řetězec "BNN^AA@A".

Tuto transformaci využívá například kompresní metoda bzip2.

5.1.3 Huffmanovo kódování

Huffmanovo kódování je algoritmus entropického kódovaní dat používaný pro bezeztrátovou kompresi. Vstupem tohoto algoritmu je sekvence symbolů (znaků, čísel), výstupem pak bitový proud dat. Algoritmus nahrazuje symboly dle četnosti jejich výskytu ve vstupním toku za sekvence bitů různé délky. Častěji se vyskytujícím symbolům jsou přiřazeny kratší bitové sekvence.

Metoda nejprve projde vstupní sekvenci symbolů a spočítá statistiku četnosti výskytu jednotlivých symbolů. Z této statistiky pak sestaví binární strom. Tvorba tohoto stromu spočívá v tom, že jsou postupně párovány symboly nebo uzly stromu s nejmenší četností. Tyto vytvoří nový uzel s hodnotou danou součtem jejich četností. Ve vytvořeném stromu je tedy nejkratší cesta od kořene k uzlům s nejvyšší četností výskytu. Ve druhém průchodu vstupních dat je pak jednoduše tvořen výstup náhradou symbolu ze vstupu odpovídající cestu ve stromu (jeden z poduzlů reprezentuje bit 0, druhý bit 1).

Tato metoda není na rozdíl od aritmetického kódování patentově chráněna. Využívá ji například kompresní metoda bzip2.

5.2 Kompresní algoritmus bzip2

bzip2 je free software/open source algoritmus komprese dat (a také program) vyvinutý Julianem Sewardem. Tento algoritmus je efektivnější než například gzip nebo ZIP. Neumožňuje uložení více souborů, ale pouze toku dat. Při kompresi používá metodu RLE, dále pokročilé metody jako BWT a Huffmanovo kódování.

5.2.1 Knihovna libbzip2

K algoritmu je vytvořena knihovna nazvaná libbzip2. Tato knihovna deklaruje pro programátora rozhraní s dvěma úrovněmi funkcí. První pracuje přímo se soubory, ale druhá s jakýmkoli polem dat uloženým v paměti. Dá se tedy použít pro kompresi toku čísel reprezentujících kvantizované koeficienty obrazu. Knihovně je možné předávat data postupně, což vyhovuje postupné kompresi po blocích 8×8 pixelů. Výstupní proud dat produkovaný touto knihovnou je možné přímo zapisovat do výstupního souboru.

6 Implementace

Cílem této práce bylo navrhnout postup ztrátové komprese obrazu s využitím pokročilých metod a implementovat knihovnu pro kompresi a dekompresi obrazu pracující na základě tohoto algoritmu.

Tato knihovna dokáže převést obraz uložený v paměti pomocí metod uvedených v předchozích kapitolách do souboru, který v ideálním případě zabírá zlomek velikosti původního obrazu. V závislosti na použité metodě a nastavené kvalitě komprese se velikost souboru zvětšuje nebo zmenšuje a zároveň se odpovídajícím způsobem zvyšuje nebo snižuje kvalita zpětně rekonstruovaného obrazu.

6.1 Postup komprese

Knihovna očekává v paměti bitmapu obrazu ve formátu RGB. Z programového hlediska je tato bitmapa jednorozměrným polem. Jak jsem vysvětlil v podkapitole 3.2.2, není z důvodu velkých výpočetních nároků možné obraz zpracovávat jako celek. Obraz je proto rozdělen na bloky 8×8 pixelů. Velikost bloku lze měnit, musí však být mocninou 2. Bloky jsou zpracovávány postupně po řádcích. Protože velikost obrázku nemusí vždy přesně zapadat do sítě bloků velikost 8×8 px, je případná oblast za okrajem doplněna pixely s nulovými hodnotami (tedy černou barvou).

Každý zpracovávaný blok je nejprve převeden z barevného modelu RGB na model $Y'C_bC_r$ a to podle rovnic (1) a (2) definovaných ve specifikaci JFIF [2]. Následně je na všech 3 složkách bloku provedena transformace na frekvenční koeficienty. Knihovna podporuje dvě transformace označované jako DCT a DWT. Obě transformace produkují reálná čísla se znaménkem. V případě DCT se jedná o klasickou diskrétní kosinovou transformaci. V případě DWT je to Haarova transformace. Transformace zpracovávají blok nejprve po řádcích a poté po sloupcích. Ze vzorců (3) a (4) v podkapitole 4.1 je patrné, že se při výpočtu neustále opakuje násobení funkcí *cos*. Funkce zodpovědné za DCT a IDCT si tyto hodnoty při prvním použití předpočítají a tím zrychlí výpočet transformace. Ukázky zdrojových kódů transformací jsou uvedeny v příloze 2.

6.1.1 Kvantizace a linearizace

Po dopředných transformacích každé složky obrazu jsou na koeficientech provedeny kvantizace. Použitá kvantizační matice závisí na požadované kvalitě komprese, kompresní metodě a složce (jiná pro intenzitu Y a jiná pro chromatické složky). Za referenční knihovna přebírá původní kvantizační matice standardu JPEG zobrazené na obrázku číslo 5. Tyto matice nepoužije ovšem přímo, ale upraví je dle požadované kvality a metody. Požadovaná kvalita je knihovně předána v procentech (celé číslo v intervalu 1 až 100). Velikost souboru ani člověkem vnímaná kvalita však lineárně neodpovídá těmto procentům. Je to pouze koeficient pro výpočet kvantizačních matic.

Pokud je požadovanou metodou DCT, použije knihovna vzorce vycházejícího opět ze standardu JPEG. Protože však tento standard používá mírně jinou metodu DCT, použije vzorec upravený. Původní vzorec je znázorněn v tabulce 2 (kvalita je značena q).

Vstupní kvalita q	$1 \leq q < 50$	$50 \leq q \leq 100$
Upravená kvalita	$5000 / q$	$2000 - q \cdot 2$

Tabulka 2: Přepočet kvality u metody DCT

Experimentováním s touto metodou jsem zjistil, že větších rozdílů mezi jednotlivými kvalitami metoda dosahuje, když bude tento koeficient vynásoben číslem 6. Tím se kvalita dostala do rozsahu 30 000 až 0. Tímto upraveným koeficientem kvality je poté vynásobeno každé číslo v kvantizační matici a výsledek podělen stem (100). Výpočet je ve skutečnosti mírně složitější. Koeficienty v kvantizační matici jsou tak ovlivněny požadovanou kvalitou.

V případě použití metody DWT je kvalita pouze obrácena do intervalu 1 až 100 (z kvality 1 procento vznikne koeficient 100). Tímto číslem je opět vynásoben koeficient v kvantizační matici a výsledek poté podělen stem.

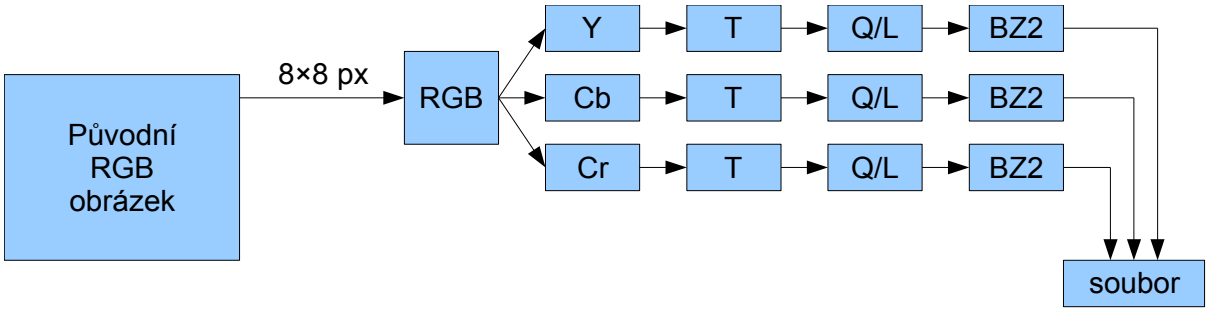
Výsledkem kvantizace je 16bitové číslo se znaménkem. Funkce provádějící kvantizaci zároveň koeficienty linearizuje. Při linearizace je nejprve záporná část převedena na kladnou. To se děje prokládáním. Hodnota kladných i záporných čísel je nejprve zdvojnásobena. Poté se záporná převedou na kladná a sníží o 1. Tím vzniká 16bitové číslo bez znaménka. Kvantizované koeficienty jsou většinou čísla blízká nule. Je proto pravděpodobné, že vyšší bajt čísla bude nulový. Tato čísla jsou poté uložena do jednorozměrného pole *zig-zag* průchodem pomocí předpočítaného pole indexů.

6.1.2 Práce s knihovnou libbzip2

Linearizovaná pole koeficientů jsou poté zkomprimována knihovnou libbzip2 a uložena přímo do výstupního souboru. Nejprve jsou takto zpracovány koeficienty ze složky Y, následně koeficienty chromatických složek. Práce s knihovnou libbzip2 je jednoduchá a dobře popsaná v manuálu [4]. Rozhraní knihovny je deklarováno v hlavičkovém souboru *bzlib.h*. Před kompresí je třeba nejprve inicializovat datové struktury voláním funkce *BZ2_bzCompressInit()*. Následuje opakované volání funkce *BZ2_bzCompress()* s parametrem *BZ_RUN*. Tím je knihovna požádána o kompresi daného bloku dat. Při této činnosti jsou předávaná data analyzována, jsou vytvářeny paměťové struktury a může, ale nemusí být vyprodukován výstup (zkomprimovaná data). Po zpracování všech bloků obrazu je nutné zavolat funkci *BZ2_bzCompress()* s parametrem *BZ_FINISH*. Tím je knihovna požádána o dokončení komprese. Při této činnosti budou vyprodukována data. Volání je nutné opakovat, dokud funkce nevrátí návratovou hodnotu *BZ_STREAM_END*. Po dokončení komprese je ještě nutné zavolat *BZ2_bzCompressEnd*() pro uvolnění datových struktur. Do toku dat předávaných knihovně libbzip2 je ještě předána informace o použité metodě komprese, kvalitě

a rozměry obrazu. Vyprodukovaný soubor je platným souborovým formátem bzip2. Lze jej tedy rozbalit a koeficienty si prohlížet.

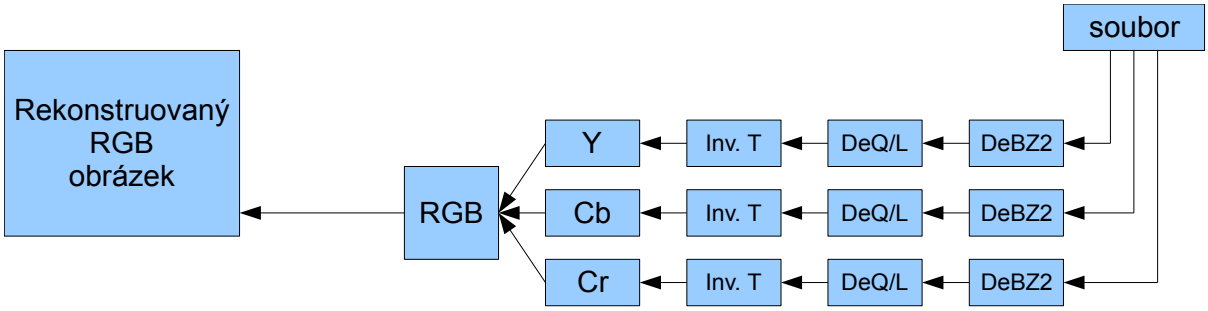
Celý postup komprese je zjednodušeně znázorněn na následujícím obrázku. T znamená dopředná transformace, Q/L kvantizace a linearizace, BZ2 komprese pomocí knihovny libbzip2.



Obrázek 7: Blokový diagram komprese

6.2 Postup dekomprese

Při dekompresi knihovna očekává deskriptor souboru se zkomprimovaným obrazem a vytváří v paměti bitmapu tohoto obrazu ve formátu RGB. Ze zkomprimovaných dat jsou nejprve dekomprimovány informace nutné k dalšímu postupu rekonstrukce obrazu (tj. rozměry, metoda a kvalita). Dle rozměrů je poté alokována paměť.



Obrázek 8: Blokový diagram dekomprese

K celé dekompresi je opět využíváno služeb knihovny libbzip2. Postup dekomprese spočívá v inicializaci funkcí *BZ2_bzDecompressInit()*, sérií volání funkce *BZ2_bzDecompress()* a nakonec dealokací struktur voláním *BZ2_bzDecompressEnd()*.

Z metody a kvality, se kterou byl obraz komprimován, jsou opět vypočítány kvantizační matice. Celý postup konstrukce obrazu probíhá inverzně ke kompresi. Bloky jsou z bzip2 toku vytahovány postupně po řádcích, jsou uloženy do jednorozměrných polí, delinearizovány, dekvantizovány a nakonec je na nich provedena inverzní transformace k transformaci použité

při kompresi. Při kompresi ovšem došlo k zaokrouhlení koeficientů. To má nyní za následek vznik artefaktů v obraze. Čím větší zaokrouhlení bylo provedeno, tím je obrázek méně kvalitní. Způsob poškození obrázku se liší také podle použité transformace. Výsledkem dekomprese je tedy obraz v paměti, který není zcela shodný s původním obrazem. Celý postup dekomprese je opět znázorněn na obrázku číslo 8.

6.3 Demonstrační programy

Ke knihovně jsem vytvořil 2 demonstrační programy. První (*png2open*) dokáže převést obrázek z bezeztrátového formátu PNG požadovanou metodou a kvalitou na můj formát OPEN. Druhý (*open2png*) z tohoto formátu zrekonstruuje obrázek a uloží jen zpět do PNG. K práci s formátem PNG používám referenční knihovnu libpng. Tato knihovna využívá ke kompresi obrazu jinou knihovnu zlib. Programy jsou napsány stejně jako knihovna v jazyce ISO C99 a nevyužívají žádné platformově závislé prostředky. Práce s demonstračními programy je popsána v příloze 4.

6.4 Srovnání metod komprese

Kvalita obrazu je do značné míry subjektivní dojem. Nelze jednoznačně prohlásit jednu metodu komprese za efektivnější (poměr kvalita/velikost souboru). Efektivnost metody závisí značně na vstupním obraze. Kvalita obrazu ani velikost souboru se nemění lineárně s parametrem požadované kvality. Metody DCT i DWT, které jsem implementoval, jsou určeny především pro fotografie (nikoli tedy pro obrázky s ostrými hranami). Pro porovnávání kvality dvou obrazů (nebo videí) se používá několik metod. Nejběžnější z nich je metoda PSNR.

6.4.1 Metoda PSNR

PSNR je zkratka z anglického *peak signal-to-noise ratio*. Termín vyjadřuje poměr mezi maximální možnou energií signálu a energií šumu. Protože mnoho signálů má velmi široké dynamické spektrum, obvykle se PSNR vyjadřuje v logaritmickém měřítku.

K definici PSNR je třeba nejprve definovat střední kvadratickou chybu (anglicky *mean squared error* neboli MSE). MSE je pro dva černobílé obrazy I a K o rozměrech m×n definována jako:

$$MSE = \frac{1}{mn} \cdot \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \left[I(i,j) - K(i,j) \right]^2$$

PSNR je poté definována jako:

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right) = 20 \cdot \log_{10} \left(\frac{MAX_I}{\sqrt{MSE}} \right)$$

MAX_I je maximální hodnota pixelu v obrázku (tedy 255). Pro RGB obrázky je MSE suma přes všechny složky(R, G, B) dělena třemi. Typická hodnota PSNR pro komprimované obrázky je mezi 30 a 40 dB.

6.4.2 Porovnání DCT, DWT a formátu JPEG

Pro porovnání jsem použil obrázek (fotografii) nepoškozenou předchozí kompresí o rozměrech 1024×576 pixelů. Fotografii jsem zkomprimoval mou knihovnou jednak metodami DCT i DWT a také pro porovnání do formátu JPEG. Porovnání jsem prováděl ve třech stupních kvality a to 30%, 50% a 85%. Vyšší kvality nemají smysl, protože při nich neúměrně roste velikost souboru a lidské oko rozdíl stejně nepozná. Porovnával jsem jednak hodnotu PSNR od originálního obrázku a také velikost souboru s komprimovaným obrázkem. Z hodnot jsem sestavil následující tabulku:

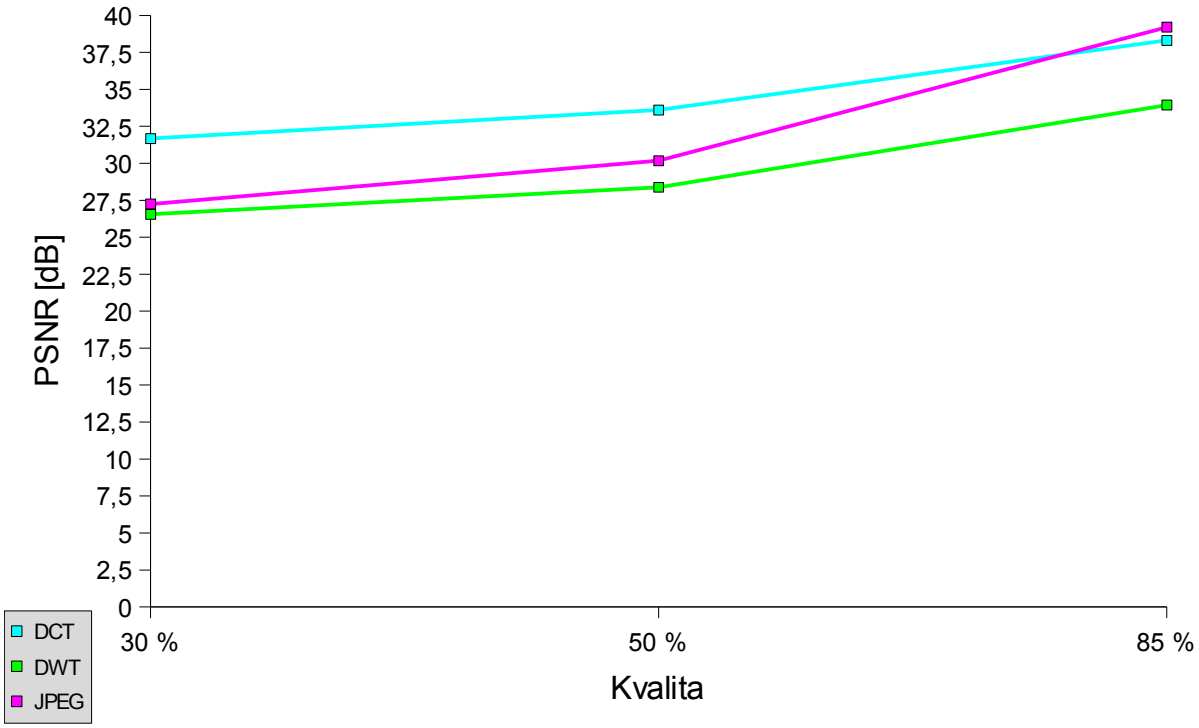
Metoda	Kvalita [%]	Velikost [B]	PSNR [dB]			
			R	G	B	RGB
DCT	30	62411	31,64526	32,03102	31,35165	31,67598
DCT	50	85937	33,58253	33,94114	33,28919	33,60429
DCT	85	167534	38,28761	38,65823	37,97959	38,30848
DWT	30	32311	26,22096	27,32083	26,10608	26,54929
DWT	50	43924	28,34499	28,97463	27,81137	28,37700
DWT	85	108261	33,86951	34,61543	33,34380	33,94291
JPEG	30	32741	27,18253	27,78385	26,76230	27,24289
JPEG	50	47248	30,15493	30,68377	29,68583	30,17484
JPEG	85	157951	39,17663	39,66368	38,78695	39,20909

Tabulka 3: Porovnání kvality komprese metodami DCT, DWT a formátu JPEG

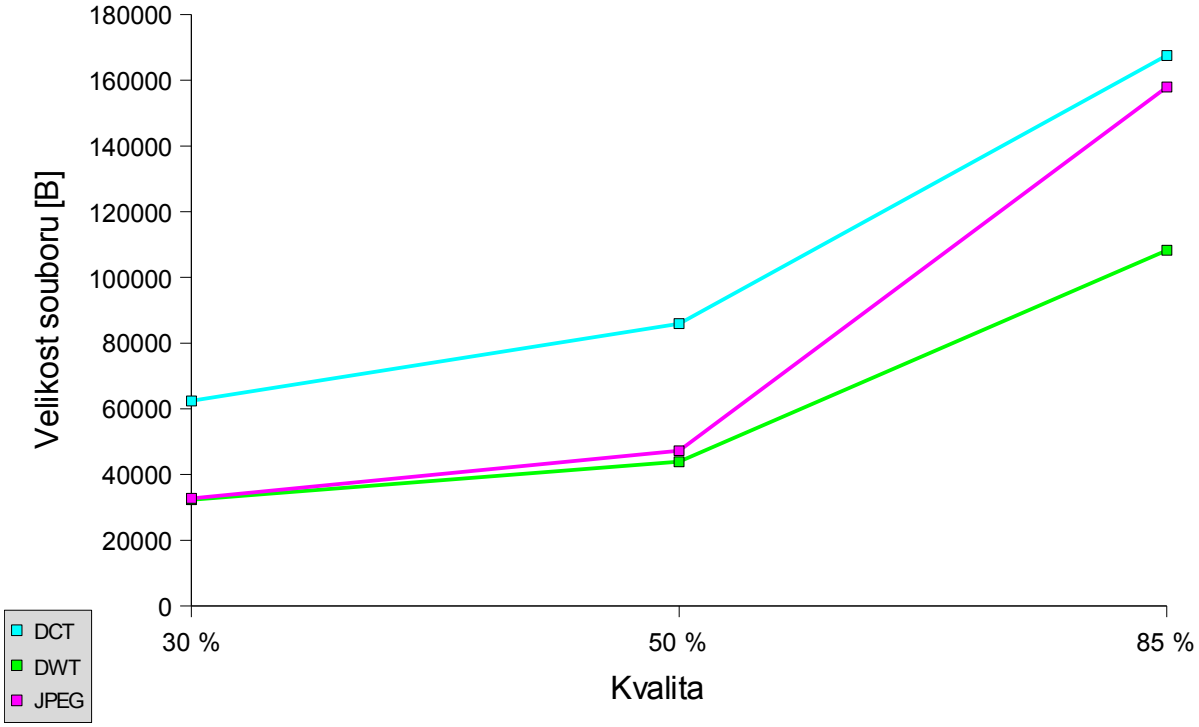
Při ukládání do formátu JPEG jsem nenastavil žádné podvzorkování chromatických složek, stejně jako to dělá má knihovna. S tabulky jsem vytvořil následující grafy. Tyto grafy jsou ovšem mírně matoucí, protože přepočet kvality na kvantizační matici jsem vytvořil experimentálně.

Z prvního grafu se zdá, že nejlepších výsledků dosahuje metoda DCT, nejhorších DWT. To je ovšem bez znalosti odpovídající velikosti souboru neúplná informace. Z druhého grafu vyplývá, že DCT má při dané kvalitě také největší velikost souboru. Velikost obrázku ve formátu JPEG s přibývající kvalitou rychle vzrůstá. Vše je ovšem dáno způsobem přepočtu procentuální kvality na použitou kvantizační matici.

Obecně lze říci, že DWT dosahuje nejnižší kvality, ale také nejnižší velikosti souboru. DCT kvality nejvyšší ovšem s největším souborem. Při 30 a 50 procentní kvalitě JPEG dosahoval velikosti souboru srovnatelné s DWT, ale kvalitu měl vyšší. U 85 procent má JPEG velikost souboru o něco nižší než má knihovna s metodou DCT a kvalitu přitom vyšší. Vzorové obrázky jsou ukázány v příloze 3.



Obrázek 9: Porovnání PSNR jednotlivých metod a kvalit komprese



Obrázek 10: Porovnání velikosti souboru jednotlivých metod a kvalit komprese

6.5 Rozhraní knihovny

Programová dokumentace je uvedena jako příloha 1. Knihovnu jsem nazval *libopen.* OPEN je zkratka z *Open Photography ExchaNge format*. K projektu jsem zřídil domovskou stránku na http://libopen.sourceforge.net/.

K překladu knihovny a demonstračních programů je potřeba knihovna libbzip2 (http://www.bzip.org/) a knihovna libpng (http://www.libpng.org/pub/png/libpng.html).

7 Závěr

Účelem práce bylo vytvořit knihovnu pro kompresi a dekompresi obrazu. Knihovnu jsem vytvořil a implementoval kompresní metody pracující na základě dvou rozdílných algoritmů. Tato knihovna je napsána multiplatformně (z hlediska hardwarového i softwarového). Při seznamování se s metodami komprese jsem o nich vytvořil několik článků na české Wikipedii. Vytvořená knihovna se jmenuje libopen, je možné ji stáhnout z http://libopen.sourceforge.net/.

Efektivnost mnou implementovaných kompresních metod je srovnatelná s nejrozšířenějším formátem JPEG. Metoda používající diskrétní kosinovou transformaci má o něco lepší výsledky než metoda používající Haarovu diskrétní vlnkovou transformaci. To je způsobeno jednak jednoduchostí zvolené vlnkové transformace a také neoptimalizovanou kvantizační maticí pro tuto metodu.

V dalším vývoji bych chtěl implementovat některou složitější vlnkovou transformaci a nalézt pro ni nejvhodnější kvantizační matici. Tato metoda komprese by se dala rozšířit z dvourozměrného obrazu na video, které lze chápat jako obraz trojrozměrný.

Literatura

[1] Charles Poynton, *YUV and luminance considered harmful: a plea for precise terminology in video*, 8. března 2001. Dokument dostupný na URL http://poynton.com/papers/YUV_and_luminance_harmful.html (8. května 2007).

[2] Eric Hamilton, C-Cube Microsystems, 1778 McCarthy Blvd., Milpitas, CA 95035, *JPEG File Interchange Format Version 1.02*, 1. září 1992. Dokument dostupný na URL http://www.jpeg.org/public/jfif.pdf (8. května 2007).

[3] ITU/CCITT, *ISO/IEC IS 10918-1 | ITU-T Recommendation T.81*, září 1992. Dokument dostupný na URL http://www.w3.org/Graphics/JPEG/itu-t81.pdf (8. května 2007).

[4] Julian Seward, *bzip2 and libbzip2, version 1.0.3: A program and library for data compression*, 1995. Dokument dostupný na URL http://www.bzip.org/1.0.3/bzip2-manual-1.0.3.pdf (8. května 2007)

[5] Charles Poynton, *Frequently Asked Questions about Color*, 2. března 1997. Dokument dostupný na URL http://www.poynton.com/PDFs/ColorFAQ.pdf (8. května 2007)

[6] Wikipedia, The Free Encyclopedia, *Discrete cosine transform*, 8. května 2007. Dokument dostupný na URL http://en.wikipedia.org/wiki/Discrete_cosine_transform (8. května 2007)

[7] Wikipedia, The Free Encyclopedia, *Discrete wavelet transform*, 8. května 2007. Dokument dostupný na URL http://en.wikipedia.org/wiki/Discrete_wavelet_transform (8. května 2007)

Seznam příloh

Příloha 1. Manuál

Datové struktury
Pixel:
```
typedef struct
{
    uint8 r,g,b;
} TPixel, *PPixel;
```
Pixel se skládá z barevných komponent RGB (8 bitů na složku).

Framebuffer:
```
typedef struct
{
    PPixel buff;
    uint32 w,h;
} TFrameBuff, *PFrameBuff;
```
Framebuffer je vlastní obrázek reprezentovaný jako jednorozměrné pole pixelů o rozměrech *w* (šířka) a *h* (výška). Mapovací funkce pole je *y*w+x*, kde *x* a *y* jsou souřadnice v obrázku.

Metody komprese:
```
typedef enum
{
    DCT = 0,
    DWT = 1,
} openMethod;
```
Výčtový datový typ určující metodu komprese. *DCT* znamená diskrétní kosinová transformace, *DWT* znamená diskrétní vlnková (waveletová) transformace.

Funkce
Komprese:
```
int openCompress(FILE *descriptor, PFrameBuff framebuffer,
openMethod method, int quality);
```
Provede kompresi dle zadané metody.
Parametry:
descriptor - otevřený soubor (w), kam zapsat zkomprimovaný obrázek
framebuffer - obrázek ve FB ke zkomprimovaní (nemění)
method - metoda komprese *DCT*/*DWT*
quality - kvalita (stupeň komprese) v % (1..100)
Vrací:
!=0 při chybě

Dekomprese:
```
int openDecompress(FILE *descriptor, PFrameBuff *framebuffer);
```
Provede dekompresi obrázku do FB (který vytvoří).
Parametry:
descriptor - soubor otevřený (r) se zkomprimovaným obrázkem
framebuffer - kam obrázek uložit (alokuje sám)
Vrací:
!=0 při chybě

Hlavičkové soubory
open.h
Veškeré rozhraní je deklarováno v tomto souboru.

Použití
Přeložte soubor *open.c* a slinkujte s vaší aplikací. Při kompilaci a linkovaní budete potřebovat knihovnu *libbzip2*. Příklad překladu je uveden v *Makefile* v adresáři *src*.

Příloha 2. Ukázky zdrojových kódů

Dopředná 1D DCT:

```
void fct(const double *input, double *output)
{
        for(int h=0; h<LENGTH; h++)
        {
                double sum = 0;
                for(int j=0; j<LENGTH; j++)
                {
                        double xk = input[j];
                        double c = (M_PI/LENGTH)*h*(j+0.5);
                        sum += xk*cos(c);
                }
                output[h] = sum;
        }
}
```

Zpětná 1D DCT:

```
void ict(const double *input, double *output)
{
        for(int h=0; h<LENGTH; h++)
        {
                double sum = 0;
                for(int j=1; j<LENGTH; j++)
                {
                        double xk = input[j];
                        double c = (M_PI/LENGTH)*j*(h+0.5);
                        sum += xk*cos(c);
                }
                sum += 0.5*input[0];
                sum *= 2/(double)LENGTH;
                output[h] = sum;
        }
}
```

Dopředná 1D DWT:

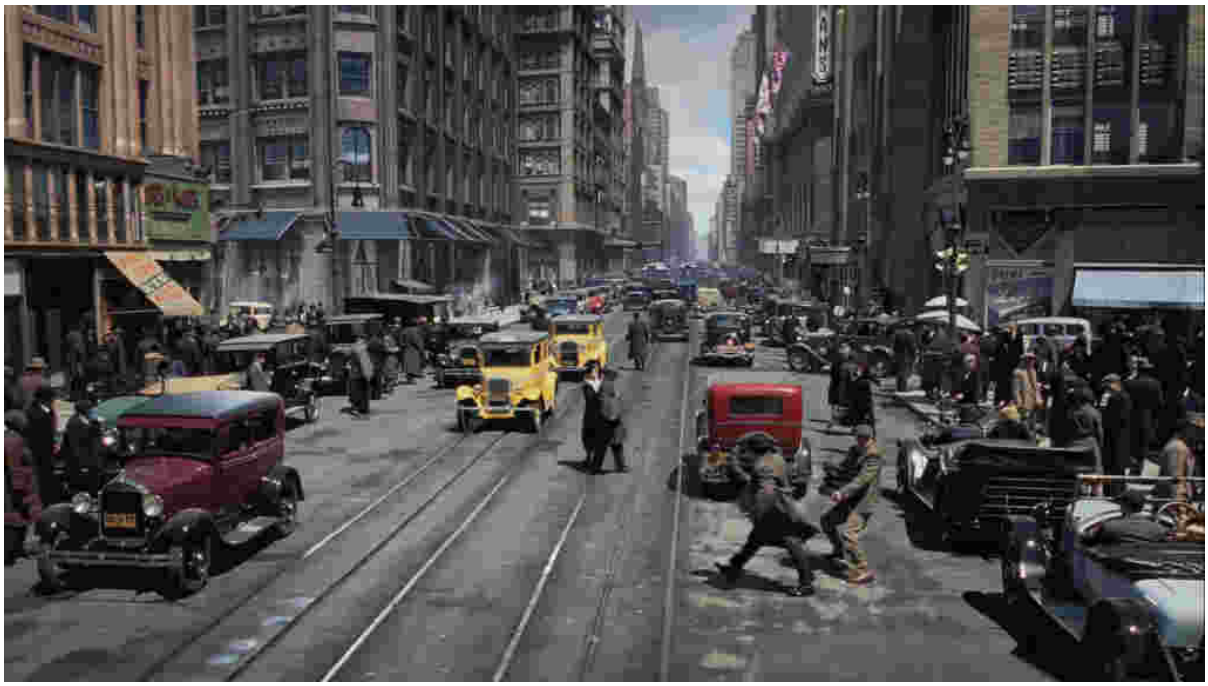
```
void fwt(const double *const_input, double *output)
{
        static double input[LENGTH];
        memcpy(input,const_input,sizeof(double)*LENGTH);
        for(int length = LENGTH >> 1; ; length >>= 1)
        {
                for(int i = 0; i < length; i++)
                {
                        double sum = (input[i*2]+input[i*2+1])/2;
                        double difference = (input[i*2]-input[i*2+1])/2;
                        output[i] = sum;
                        output[length+i] = difference;
                }
                if(length == 1)
                        return;
                memcpy(input,output,sizeof(double)*length);
        }
}
```

Zpětná 1D DWT:

```
void iwt(const double *const_input, double *output)
{
	static double input[LENGTH];
	memcpy(input,const_input,sizeof(double)*LENGTH);
	for(int length = 1; ; length <<= 1)
	{
		for(int i = 0; i < length; i++)
		{
			double a = (input[i] - input[length+i]);
			double b = (input[i] + input[length+i]);
			output[i*2] = b;
			output[i*2+1] = a;
		}
		if(length == LENGTH >> 1)
			return;
		memcpy(input,output,sizeof(double)*length*2);
	}
}
```

Příloha 3. Ukázkové obrázky

Obrázek 11: Původní obrázek



Obrázek 12: JPEG s kvalitou 30%

Obrázek 13: Má implementace DCT s kvalitou 30%

Obrázek 14: Má implementace Haarovy DWT s kvalitou 50%

Příloha 4. Demonstrační programy

Na přiloženém CD jsou jak zdrojové kódy demonstračních programů tak zkompilované spustitelné soubory pro některé platformy včetně Microsoft Windows. Na tomto CD je i několik vzorových obrázků, na kterých je možno práci knihovny vyzkoušet.

png2open

Popis:

Konvertuje obrázek z formátu PNG do formátu OPEN zadanou kvalitou a metodou.

Použití:

`png2open <vstup.png> <vystup.open> <metoda> <kvalita>`

Parametry:

<*metoda*>: `dct` nebo `dwt`

<*kvalita*>: `1` až `100`

Příklad:

```
png2open fruits.png fruits-dct-20.open dct 20
```

open2png

Popis:

Konvertuje obrázek z formátu OPEN na PNG.

Použití:

`open2png <vstup.open> <vystup.png>`

`Příklad:`

```
open2png fruits-dct-20.open fruits-dct-20.png
```